SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NRL Report 9049	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Research Laboratory 6b. OFFICE SYMBOL (*If applicable*) Code 5110	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (*City, State, and ZIP Code*) Washington, DC 20375-5000	7b. ADDRESS (*City, State, and ZIP Code*)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Naval Oceanographic Office 8b. OFFICE SYMBOL (*If applicable*) Code 8311	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*) NSTL Station Bay St. Louis, MI 39522	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO. DN155-707
11. TITLE (*Include Security Classification*) Software Design for an Airborne Gravity Measurement System	
12. PERSONAL AUTHOR(S) Peters, Mary F., Brozena, John M., and Clamons, J. Dean	
13a. TYPE OF REPORT Final 13b. TIME COVERED FROM 7/1/86 TO 2/28/87 14. DATE OF REPORT (*Year, Month, Day*) 12 August 1987 15. PAGE COUNT 36	
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	Gravity, Geodesy, Remote sensing, Software

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

This report describes the results of a software design study for an airborne gravity measurement system. The software is intended for data acquisition and monitoring aboard a fixed-wing aircraft over oceanic areas, and for the postflight processing required to compute gravity along a flight track. Various requirements of the postflight processing system are discussed. A method to compute gravity from airborne measurements is outlined; as some quantities can be derived from more than one measurement, alternative algorithms, to be used in cases of poor quality or missing data, are given.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Mary F. Peters	22b. TELEPHONE (*Include Area Code*) (202) 767-2024 22c. OFFICE SYMBOL Code 5110

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

☆U.S. Government Printing Office: 1985—507-047

Naval Research Laboratory
Washington, DC 20375-5000

NRL Report 9049

Software Design for an Airborne Gravity Measurement System

MARY F. PETERS, JOHN M. BROZENA, AND J. DEAN CLAMONS

Acoustics Media Characterization Section
Acoustics Division

August 12, 1987

CONTENTS

SOFTWARE DESIGN FOR AN AIRBORNE GRAVITY MEASUREMENT SYSTEM

INTRODUCTION

This report is a software design study for the acquisition and reduction of data from a fixed-wing airborne gravity measurement system (AGMS). This report follows a hardware design study [1]. The proposed system is designed for use over open water, sea-ice covered regions, coastal areas, or any regions where accurate absolute altitudes (radar or laser over a point of known altitude) are periodically obtainable. The information in this report is primarily based on experience gathered in acquiring and analyzing the data from a prototype AGMS at the Naval Research Laboratory (NRL) since 1980 [2-5]. During this program we have accumulated over 800 flight hours experience testing various hardware and software designs and modifications. The data acquisition software proposed in this report is based on the hardware proposed in Ref. 1, while the data reduction methods are more general in nature and assume only sufficiently sampled time-tagged records of the various types of input data.

Accurate knowledge of the spatial variations of Earth's gravity field is required for several reasons. Among these are compensation of inertial guidance systems, improvements in gravity models used in orbital calculations, crustal density determination in regions of known topography, and estimation of topographic structure in oceanic areas with sparse bathymetric data. Measurement of gravity variations in oceanic areas is conventionally performed from surface ships or submarines, but the greater speed and range and lower cost per track kilometer of fixed-wing aircraft make them attractive alternative platforms. The major difficulty in performing airborne gravity measurements is that of separating the variations in gravity from the vertical accelerations of the aircraft. This problem has prevented aircraft from becoming the measurement platforms of choice in the past.

Vertical acceleration of an aircraft can arise from variations in altitude above a reference surface and from the centripetal acceleration caused by motion at a constant altitude over the curved reference surface. The first type of vertical acceleration can be determined by accurate measurement of a time series of altitudes above the reference, and the second can be determined by precise navigation. The three parameters necessary for airborne gravity measurement are total vertical acceleration, the sum of gravity and the vertical accelerations caused by the motion of the aircraft; altitude, for determination of the vertical accelerations of the aircraft caused by variations in altitude as well as the correction for the vertical attenuation of gravity with height; and navigation, to determine the centripetal acceleration or Eotvos correction and the free-air normal gravity.

Total vertical acceleration is measured by a gravity meter, altitude is determined from a combination of radar and pressure altimeters, and navigation is provided by the satellite Global Positioning System (GPS). Data from these sources must be collected, time tagged, and stored aboard the aircraft for later data reduction and analysis.

Manuscript approved May 11, 1987.

REAL-TIME DATA ACQUISITION AND MONITORING

General Program Design Considerations

The real-time gravity data acquisition and monitoring system is divided into modules that, to the maximum degree possible, operate independently of each other. This modularity guarantees that the failure of one module (whether the failure is in hardware or in software) has minimal impact on the rest of the system and that a change of hardware requires changes in only a few modules. Modularity in the system is achieved by having separate programs for handling different hardware components of the system and by writing separate data files for different types of data.

In specifying a system that consists of a number of programs running simultaneously, we are making some assumptions about the underlying operating system. The operating system must support multitasking and should be interrupt driven. That is, the switching of tasks should be triggered by interrupts caused by external events (e.g., a block of navigation data is ready). The system is also assumed to support direct memory access data transfers.

Most of the system can be written in a high-level language (we have used FORTRAN). There are, however, some parts that will require low-level programming. As will be seen later, some special input/output (I/O) driver requirements must be satisfied. These may require user-written drivers, which usually implies using assembly language.

Throughout the system care must be taken to accurately tag each piece of data with time. The subsequent processing of the data requires recombining the data in the various files and on performing time series analysis. This can only be done if the time of acquisition of every data point can be determined. There is one oscillator in the system that determines sample rates and drives the system time-of-day clock. This clock should be read by each device driver when data are ready, to ensure that the operating system overhead does not interfere with time accuracy.

It is extremely desirable to have a program that monitors and displays the inputs from the various sensor systems in real time. This program should also perform a rough gravity estimation to evaluate the operation of the system. The monitor program requires a method of interprogram communication. In many systems this is accomplished by a shared memory block. Some operating systems may require that programs send messages to each other. The method of implementation is not of primary importance, but the operating system must provide some method of interprogram communication.

The data from the gravity meter, radar altimeter, pressure altimeter, and navigation subsystems are placed in separate data files. This is done largely for reasons of modularity mentioned above. This organization also allows for relatively simple data editing programs in the preliminary processing of the data. For the most part we have decided to use binary, fixed record length files. This is a compact method of data storage that does not require time-consuming conversion of format for processing. It also allows random file access techniques to be used that greatly speed access to particular sets of data. The disadvantage to using binary files is that they are not directly readable by humans. Because of this it is necessary to have good display and editing capabilities for binary files. The exception to this raw data file format is that the navigation data files contain ASCII records. This decision was made because the volume of navigation data is smaller than other data and they need to be edited more frequently. For each data type, the data are broken up into files of about one hour in duration. This means that if a disk error occurs it is likely to affect only one data file, and only one hour of data is lost.

Data Acquisition Programs

Master Control Programs—AGSS and PARAM

For operator convenience, one program has the responsibility to start all the other programs in the system. This program, which we call AGSS, first runs an initialization program, PARAM, which asks the operator for some information about operating parameters. These parameters are sent to the acquisition and monitoring programs for their use. PARAM is designed as a separate program so that it may be run at any time during a flight to alter the operating parameters. AGSS then sets an operating system flag that means that AGSS is running. The nature of this flag varies according to the operating system, but the important feature is that other programs can determine the value of this flag. They use the flag to determine when to close data files and stop acquiring data. AGSS proceeds to start all of the data acquisition and monitoring programs: RDGRV, RDALT, RDGPS, and GRMON. After this, AGSS has no responsibility except to find out when the operator wishes to stop collecting data. It then resets the AGSS running flag and stops.

The outline of AGSS is

```
Program AGSS
    Run PARAM to get operating parameters
    Wait for PARAM to stop
    Set the AGSS running flag to TRUE
    Start the program RDGRV
    Start the program RDALT
    Start the program RDGPS
    Start the program GRMON
    Do until operator interrupts AGSS
        Suspend for one second
    Enddo
    Set the AGSS running flag to FALSE
End
```

The outline of PARAM is

```
Program PARAM
    Ask the operator for the settings of the radar altimeter
      (These values are    Time range setting
                           Multiplier setting
                           Delay setting
                           Offset owing to system delays)
    Ask the operator for the ground pressure to use in calculating
      altitude from pressure readings
    Send these values to the acquisition and monitor programs
End
```

Gravimeter—RDGRV

The rest of the description in this section assumes some familiarity with the hardware described in Ref. 1. In particular, we assume that all of the data are adequately buffered in hardware so that the software can assume that the next data point read from the hardware is the next sequential data point. We assume that no data are lost in this process. However, it is good practice to check that no data are lost and to flag data records that follow missed data.

The Lacoste-Romberg (L-R) gravity meter produces 10 analog signals and 2 digital signals that must be recorded. The analog signals are

- Average beam
- Total cross coupling
- Total correction
- Vertical acceleration squared
- Average cross acceleration
- Average longitudinal acceleration
- Heading A
- Heading B
- Zero calibration
- Raw Beam.

The digital signals and 9 of the analog signals may be recorded at 1 Hz, but the raw beam analog signal should be recorded at 10 Hz. The analog sampling is done by a scanning analog-to-digital (A-D) converter that samples all 10 channels at 100 Hz (i.e., each channel is sampled at 10 Hz) and then discarding 9 of the 10 samples for all channels except the raw beam. Resolution required is 14 bits plus a sign bit. Table 1 displays the contents of each gravity record.

Table 1

Variable Name	Type	Resolution/Scaling
Time	3 Integers	10 ms
Year	Integer	Yr
Spring tension time	3 Integers	10 ms
Average beam	Integer	Unscaled V × 3276.8
Cross coupling	Integer	Unscaled V × 3276.8
Total correction	Integer	Unscaled V × 3276.8
Vertical acceleration	Integer	Unscaled V × 3276.8
Average cross acceleration	Integer	Unscaled V × 3276.8
Average long acceleration	Integer	Unscaled V × 3276.8
Heading A	Integer	Unscaled V × 3276.8
Heading B	Integer	Unscaled V × 3276.8
Zero check	Integer	Unscaled V × 3276.8
Gravity	Floating point	0.1 mGal
Spring tension	Floating point	0.1 mGal
Raw beam 1	Integer	Unscaled V × 3276.8
.	.	.
.	.	.
.	.	.
Raw beam 10	Integer	Unscaled V × 3276.8

The first time in the record is the time that the last sample in the record from the A-D converter was read. Time is measured from the beginning of the year. Note that, with the exception of the gravity and spring tension readings, the time associated with any value in the record can be derived from the time when the last sample was taken and the known sample rate. The values that are of type "integer" are assumed to be 16 bits long. This means that to get 10 ms resolution on time, we must record three integer values. The A-D converter produces a 15 bit value that is proportional to volts. The factor 3276.8 indicated in Table 1 is correct if the A-D converter is operating in a +10 to −10 V range with the most significant bit of the converter output placed in the most significant bit of the computer word. The outline of RDGRV follows.

```
Program RDGRV
  Set filter constants for spring tension filter
  Open the first gravity data file
  Do until the last A-D sequential channel is read
    Read one channel from the A-D converter
  Enddo
  Do until AGSS running flag is FALSE
    Read 1 s of data from A-D converter and read the time at which
          the last channel was read
    Read the gravity and spring tension values and the time at which
          the spring tension was read
    Filter the spring tension
    Put the data into the output record
    If the current data file is full, then
      Close the current data file and
      Open the next data file
    Endif
    Write a record to the data file
    Send cross coupling, average beam, filtered spring tension, and
          the record time to the monitor program
    If the program is not in sync with the A-D scanner, then
      Do until the last A-D sequential channel is read
        Read one channel from the A-D converter
      Enddo
    Endif
  Enddo
  Close the gravity data files
End
```

One of the values needed to do real-time gravity estimation is spring tension. Because of the way the L-R gravity meter filters various signals, it is necessary to filter the spring tension value to match the average beam and cross coupling signals. The filter used is a three-stage low-pass resistance-capacitance (R-C) filter with a time constant of 20 s. We choose to filter the spring tension value here although it could be done by the monitor program.

It is assumed that there is a way to distinguish the last channel in the A-D converter scan sequence. This is important to ensure that the program is in sync with the A-D scanner. To be safe it is a good idea to check that it is still in sync at every record.

The Bell gravity meter presents a simpler set of data. Once per second the meter produces a count that is proportional to the measured gravity at that time. This means that we can produce a much simpler data file. Table 2 shows the format of the Bell gravity meter data records.

Table 2

Variable Name	Type	Resolution
Time	3 Integers	10 ms
Year	Integer	Yr
Count	Integer	5 mGal

The outline of RDGRV for the Bell meter is

```
Program RDGRV
  Open the first gravity data file
  Do until AGSS running flag is FALSE
    Read one value from the Bell meter and read the time at which
        the last channel was read
    Put the data into the output record
    If the current data file is full, then
      Close the current data file and
      Open the next data file
    Endif
    Write a record to the data file
    Send the gravity count and the record time to the monitor program
  Enddo
  Close the gravity data files
End
```

Pressure Altimeter—RDGRV

The output of the pressure altimeter is an analog signal. Rather than requiring another A-D converter in the system, we use the same one used by the gravity meter. Because of this it is convenient for the gravity meter acquisition program, RDGRV, to handle the pressure altimeter as well. The pressure altimeter data are sampled at 10 Hz and placed in separate files. The pressure altimeter output must be filtered before the altitude can be used by the monitor, so the same filter subroutine is used that is used for the spring tension signal. Along with the time and the pressure readings, the ground pressure is stored in each record. This value can be entered by the operator using the program PARAM, or an update may be estimated by the monitor program. Since a flight is likely to be long enough to move through a number of weather regimes, the ground pressure will probably change. It can be estimated by comparing pressure altitudes to radar altitudes during the flight.

Table 3 displays the contents of each pressure record.

Table 3

Variable Name	Type	Resolution or Units
Time	3 Integers	10 ms
Year	Integer	Yr
Estimated ground pressure	Floating point	mbar
Pressure 1	Integer	Unscaled V × 3276.8
.	.	.
.	.	.
.	.	.
Pressure 10	Integer	Unscaled V × 3276.8

The following revised outline for RDGRV for the L-R gravity meter handles the pressure altimeter. The Bell meter program would be similarly modified.

```
Program RDGRV

 Set filter constants for spring tension filter
 Set filter constants for pressure filter
 Open the first gravity data file
 Open the first pressure altitude file
 Do until the last A-D sequential channel is read
   Read one channel from the A-D converter
 Enddo
 Do until AGSS running flag is FALSE
   Read 1 s of data from A-D converter and read the time at which
         the last channel was read
   Read the gravity and spring tension values and the time at which
         the spring tension was read
   Filter the spring tension
   Do for each pressure reading
     Convert the reading to altitude
     Filter the altitude
 Enddo
   Put the gravity data into the gravity output record
   Put the pressure data into the pressure output record
   If the current gravity data file is full then
     Close the current gravity data file
     Open the next gravity data file
   Endif
   Write a record to the gravity data file
   If the current pressure file is full then
     Close the current pressure data file
     Open the next pressure data file
   Endif
   Write a record to the pressure data file
   Send cross coupling, average beam, filtered spring tension, the time of gravity
         record, filtered pressure altitude, and the time of the pressure record
         to the monitor program
   If the program is not in sync with the A-D scanner then
     Do until the last A-D sequential channel is read
       Read one channel from the A-D converter
     Enddo
   Endif
 Enddo
 Close the gravity data files
 Close the pressure data files
End
```

Radar Altimeter—RDALT

The radar altimeter produces samples at a rate of 10 kHz. These samples are averaged in the interface to produce about 100 samples per second. Since the altimeter readings are subsequently used to calculate vertical acceleration by double differentiation, the time that each sample is taken

must be known quite accurately. Since the sample rate is known, this would ordinarily not be a problem. However, if the aircraft is over ice or land, then the altimeter may not receive echos for all outgoing pulses and, therefore, will not produce an output for those pulses. This means that the samples received by the computer may not be equally spaced. To compensate for this the interface supplies a clock value at the same time as each average altitude. Table 4 displays the contents of the radar altimeter data records.

Table 4

Variable Name	Type	Resolution
Time	3 Integers	10 ms
Year	Integer	Yr
Radar offset	Floating point	ns
Radar delay setting	Floating point	ns
Radar multiplier setting	Integer	Unit-less
Radar time range setting	Integer	ns/V
Altimeter reading 1	Integer	V × 3276.8
Altimeter counter 1	Integer	100 ms
.	.	.
.	.	.
.	.	.
Altimeter reading 100	Integer	V × 3276.8
Altimeter counter 100	Integer	100 ms

The radar offset compensates for the delay in the altimeter wiring and the position of the radar antenna. The radar delay, multiplier, and time range are all settings on the radar altimeter that must be known to calculate altitude, but that are not computer readable values. They must be entered by the operator by using the program PARAM each time the settings are changed. The altimeter produces a voltage that is proportional to the time between a transmitted pulse and its received echo. This value may be converted to altitude in meters by the following formula

$$\text{Altitude} = (\text{Voltage} \times \text{Timerange} \times \text{Multiplier} + \text{Delay} - \text{Offset}) / (2 \times C), \quad (1)$$

where C is the speed of light in m/ns.

The radar altitude needs to be filtered before use in the monitor program, and we choose to do this in RDALT. To reduce the amount of calculation required, the radar voltages are filtered, and then the output of the filter is converted to altitude. This means that whenever the radar altimeter settings are changed, the filter must be restarted since this causes a jump in the voltage.

The outline of the radar altimeter program, RDALT is

```
Program RDALT
  Set filter constants
  Open the first radar altimeter file
  Do until AGSS running flag is FALSE
    Read 100 averaged altimeter voltages and clock values and the time
        of the last reading
    Put the data into the output record
```

```
    If the current radar altimeter data file is full then
      Close the current radar altimeter data file
      Open the next radar altimeter data file
    Endif
    Write a record to the current radar altimeter data file
    If the radar altimeter settings have changed then
      Reinitialize the filter settings
    Endif
    Do for each altimeter reading
      Filter the reading
    Enddo
    Convert the last filter output to altitude in meters
    Send the filtered altitude and the time of acquisition to the monitor program
  Enddo
  Close the radar altitude file
End
```

Global Positioning System Navigation—RDGPS

We assume that the source of navigation is the Texas Instruments TI-4100 GPS receiver. Its output is RS-232 and consists of several types of blocks of data. For a detailed description of the various blocks and their formats, see Ref. 6. We will use the relative navigation block of data that includes latitude, longitude, altitude, velocity north, velocity east, and vertical velocity.

The first word of each record coming from the TI-4100 is an identifier. We look only for records whose identifier is 6, which means this is a relative navigation record. The next value in the record is the length of the data portion of the record; this number should be 194 for relative navigation records. The next four values in the record give the time of this record. Reference 6 contains the details on the format and meaning of these times. As seen from Ref. 6, we need to record the following variables: OUTPOS, OUTVEL, OUTTIM, OUTFTF, APDOP, ATBIAS, APRNID, ACNVRG, ANSV, and AMODE. The first four of these give us the position, the velocity vectors, and the time. The rest of the values give us an idea of the quality of the fix.

Table 5 contains the format of the output navigation file records.

Data is sent from the TI-4100 at 9600 baud, so to avoid loss of data we must be sure that there is adequate buffering in the system to allow processing of the data stream. The data stream is mostly binary numbers with a few ASCII characters marking the beginning and end of blocks. The beginning of a block is marked by the ASCII characters DLE STX, and the end of the blocks are marked by DLE ETX. Any DLE character that occurs in the data stream is always followed by a second DLE character; therefore it is a simple matter to find the beginning of a block — just look for DLE STX. The integer values in the data stream are 2's complement 16- or 32-bit values (Ref. 6). The floating point values are 32- or 64-bit IEEE (Institute of Electrical and Electronics Engineers) format values. If the host machine does not use these formats, then conversion routines must be applied before writing the data to data files.

Table 5

Variable Name	Type	Resolution or Units
Computer system time	3 Integers	10 ms
Year	Integer	Yr
Current fundamental time frame	Double precision integer	20 ms
Fundamental time frame that matches GPS time in next field	Double precision integer	20 ms
GPS time	Double precision integer	20 ms
GPS week	Integer	1 Wk
Latitude	Double precision floating point	Rad
Longitude	Double precision floating point	Rad
Altitude	Double precision floating point	m
Velocity east	Double precision floating point	m/s
Velocity north	Double precision floating point	m/s
Velocity up	Double precision floating point	m/s
User epoch time of reference	Double precision floating point	20 ms
Fundamental time frame of applicability	Double precision floating point	20 ms
Position dilution of precision	Double precision floating point	Unitless
Initial time set flag	Integer	1 = time is set
Initial time bias available	Integer	1 = bias has been found
Id's of tracked space vehicles	4 Integers	
Solution convergence status	Integer	0 = converging, 1 = converged, 2 = diverging
Number of tracked space vehicles	Integer	
Navigation mode indicator	Integer	1 = full 3-D, 2 = altitude hold, 3 = time bias rate hold, 4 = both 2 and 3, 5 = dead reckoning

The outline of the navigation program, RDGPS, is

```
Program RDGPS
  Open the first navigation data file
  Do until AGSS running flag is FALSE
    Bodyid = 0
    Do until Bodyid = 6
      Do until DLE STX is found
        Read bytes from the TI-4100
      Enddo
    Read 2 more bytes
    Bodyid = the integer value corresponding to the bytes just read
  Enddo
 Get the current system time
  Read 404 bytes (i.e., read the rest of the record)
  Extract and convert to host machine internal format:
      Current FTF, FTF matching GPS time, GPS time, GPS week,
          OUTPOS, OUTVEL, OUTTIM, OUTFTF, APDOP, ATSET, ATBIAS, APRNID,
          ACNVRG, ANSV, and AMODE
  Put these values into the output record
  If the current navigation data file is full then
    Close the current navigation data file
    Open the next navigation data file
  Endif
  Write a record to the current navigation data file
  Send the position and velocity vectors and the time to the monitor program
 Enddo
 Close the current navigation data file
End
```

Data Quality Monitor—GRMON

The real-time monitor program for the airborne gravity system needs measurements from the other acquistion programs to calculate gravity. These values should be sent to the monitor from the other programs; a mailbox scheme in which the other programs put their data in a specified place in shared memory for the monitor to read later works well. The items that the monitor needs are

- radar altitude in meters and time for that altitude,

- spring tension, cross coupling, and average beam position in mGal with the time of these measurements,

- pressure port altimetry in meters with time along with latest estimate of sea-level pressure in mbars,

- GPS north and east velocities in m/s with time, and

- GPS latitude and longitude in degrees with time.

The units are suggested and may be changed but care must be taken that the monitor and the other programs agree on what is sent so that appropriate scales and conversions can be used. The run time

and display time spacings for the monitor can be set by the operator. They should be set so that the monitor can collect all the data needed to calculate gravity before the data from any subsystem would normally advance past the time for the next scheduled calculation (Fig. 1). Under these conditions, if the data time read is later than the next time gravity is to be displayed, part of the acquisition system has fallen behind or failed and this fact can be announced on the system console. Table 6 shows the approximate rates at which the data were sent to the monitor the last time the system was employed.

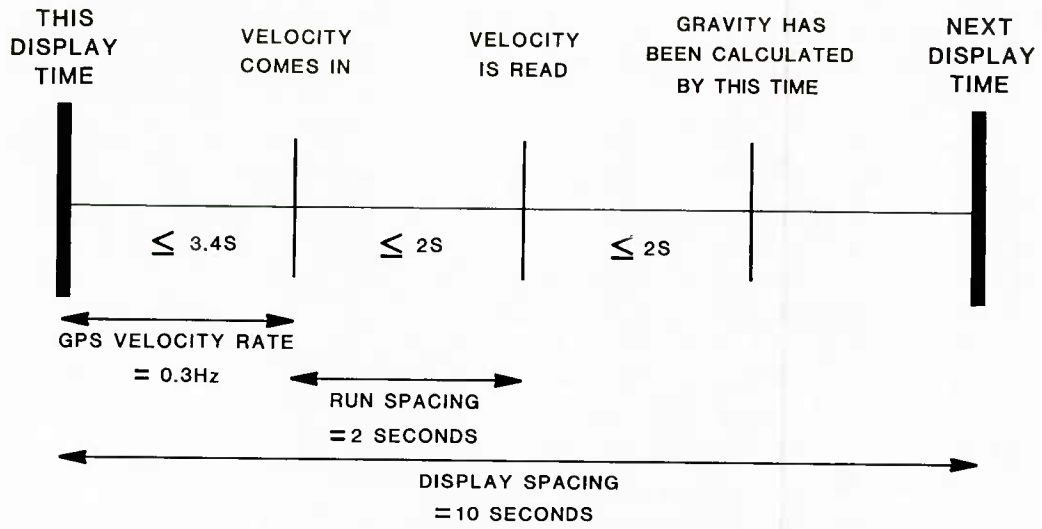


Fig. 1 — Data quality monitor timing parameters. The limiting factor on the run time and display spacing is the rate of the slowest data series acquired. In this example, GPS velocity is the slowest with a rate of 0.3 Hz.

Table 6

Radar altimetry	0.78 Hz
Pressure altimetry	1.00 Hz
L-R meter	1.00 Hz
GPS velocity	0.30 Hz
GPS positions	1.67 Hz

The run time spacing was 2 s and the display spacing was 10 s.

There are advantages to having a separate display, apart from the system console, dedicated to the monitor. Figure 2 shows what the monitor display looks like when all the data are arriving properly.

The monitor program needs to store incoming data until it has enough points to take the second derivative of altitude to get vertical acceleration. There should be a storage array with six spaces for every data type, including the time. New data points should be placed in position six; they are moved down the list as new points are read. The values written to the display are the ones from position 3. The monitor assumes that spring tension and altimetry are filtered to match beam position before being sent.

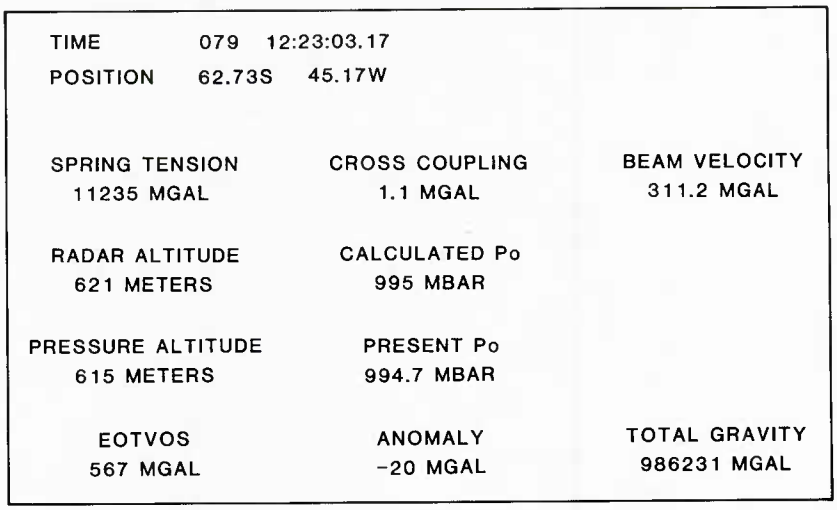


Fig. 2 — Data quality monitor display screen

Program GRMON

```
Get the timing parameters: run_time_spacing and display_spacing
Get gravity meter tie-in value grav_add
Initialize data_received flags to false and storage arrays to zero
Read a full set of data values and times from mailbox
        into current_values (NOT the storage arrays)
Time_to_display = GPS velocity time + display spacing
Do until the AGSS run flag is FALSE —
  Set past_values = current_values
  Read full set of current_values from mailbox
  Do for each type of data in: (radar, pressure, gravity meter,
        GPS velocities, GPS positions)
    If current time > time_to_display + display_spacing then Trouble = true
    Else
      Interpolate data value at time_to_display
      Place interpolated values in storage array position six
      Set data_received flag to true
    If working on GPS positions then
      Calculate model field for latitude (Eq. 2)
      Filter model field
      Store in position 6 in appropriate array
    Endif
  Endif
Enddo
If Trouble then
  Put the values that have come in on screen
  Announce what failed to show up
  Reinitialize everything like at the beginning of the program
  Read a set of current values
Else
  If all data_received flags are true then
    Set data received flags to false
    Calculate Eotvos correction (Eq. 3), filter, and put in storage arrays
```

```
        Bump storage array values down one position
        If storage arrays have been filled then
          Vertical acceleration = second derivative of radar altitude
          Calculate correction for altitude (0.3086 mGal/m)
          Compute beam velocity = derivative of beam position
          Compute sea-level pressure from radar and pressure altitudes (Eq. 4)
          Anomaly = spring tension(3) + beam velocity + cross coupling (3) + Eotvos(3)
              + altitude correction + grav_add − vertical acceleration
              − model field at latitude(3)
          Total field measured = anomaly + model field at latitude (3)
          Output values to screen
        Endif
      Endif
    Endif
    Wait run_spacing seconds
  Enddo
  END
```

The formulas referenced in the above algorithm are

$$\text{IAG67 model gravity} = 978.03185\ (1 + 0.005278895 \sin^2 L + 0.000023462 \sin^4 L), \tag{2}$$

where

L = latitude, and

$$\text{Eotvos Correction} = \frac{V_n^2}{a}\left[1 + \frac{h}{a} + f\ (2 - 3 \sin^2 L)\right] + \frac{V_e^2}{a}\left[1 + \frac{h}{a} - f \sin^2 L\right] \tag{3}$$

$$+ 2\ V_e\ \Omega \cos L \left(1 + \frac{h}{a}\right) \text{ (see Ref. 7)},$$

where

f is flattening of Earth
a is semimajor axis of Earth
h is height above the ellipsoid
V_n is velocity north
V_e is velocity east
Ω is Earth's rotation rate,

and calculated sea-level pressure CP_o is

$$CP_o = [8.4228806 \times 10^{-5} \times (Ar - Ap) + EP_o^{0.190284}]^{1/0.190284}, \tag{4}$$

where

EP_o is estimate of sea-level pressure
Ar is radar altitude
Ap is pressure altitude

GRAVITY DATA PROCESSING

General Requirements

Some general hardware and system software requirements for the processing system need to be discussed before presenting the specific algorithms used to reduce airborne gravity data. The raw data files as recorded by the acquisition system must be made available and appropriate hardware and software are needed to recover these files from storage media. Processing is easiest if the raw and processed data files can all be disk files, this requires a rather large amount of disk capacity. Gravity can be processed one survey line at a time, but since survey lines do not in general correspond nicely with the start and end times of the raw files, it is more efficient to process all the tracks from one flight at a time. Thus it is necessary to have sufficient disk space for all the data files from one flight but not the entire survey. The amount of space required is the size of all the raw files plus the size of the processing files; this will be discussed later.

The proposed processing programs have to access both the raw and processed data files. These programs should be written in a high-level language, like FORTRAN or PASCAL. The language and compiler should support binary, fixed record length, direct access disk files. Other programs are needed to display and edit the data series, the system should include a graphics applications software package and graphics hardware, including a paper plotter and a terminal with graphics capabilities. A good text editor is needed to support programming efforts, and if data files have the proper formats or format conversion programs are written, this editor can also be used for the data files.

In accordance with the hardware and software capabilities of the processing system, programs should be written to do the following basic tasks as tools for airborne gravity data reduction.

- Navigation Plotting Package — A program should exist to plot from the raw GPS position files in one of several map projections by using specified boundaries and scales. Useful projections include Mercator and polar stereographic projections. Data should be annotated by time. Another useful program to detect errors in the navigation files is a program that plots the time series of latitude and longitude vs time.

- Data Series Plots — A program should exist to plot various data series alone or in combination from the raw or processed data files; this can be used to compare different data series.

- List Raw Gravity Files — Programs should exist to list raw data records for examination. These programs should convert internal time tags to a reasonable format such as day, hour, minute, and second and rescale recorded quantities to mGals, mbars, or other appropriate units.

Raw Data Processing and Analysis Programs

Creating Merged Data Files—MKMDF

After the raw data files have been recovered from storage and placed on disk, plotted, and edited, the recorded raw data records are used to produce clean time series of all measured data types and subsequently derived values. These data series are then used to calculate gravity, and they should be evenly spaced in time. The raw data are oversampled, 2 s is an adequate sampling rate for the series used to calculate gravity. Also some series requires filtering. Average beam position, cross coupling, and all the other analog voltage outputs from the L-R meter have been internally filtered with three stages of 20 s resistance-capacitance (R-C) filtering. Counts from the Bell meter are also filtered internally with 0.75 s time constant R-C filter. This introduces different phase lags at

different frequencies, and to keep data alignment in time, all the other series should be digitally filtered to match. Such evenly spaced, filtered data series are stored in a merged data file (MDF). Table 7 shows the data series that are stored in the MDF.

Table 7

Variable Name	Type
Time: hour, s	2 Integers
Year	Integer
Latitude: deg, min	Integer, Real
Longitude: deg, min	Integer, Real
Model gravity field	Real
Eotvos	Real
GPS velocities: north, east and up	3 Reals
GPS accelerations: north, east and up	3 Reals
GPS altitude	Real
Position Dilution of Precision (PDOP)	Double precision Real
Number of satellites	Integer
Radar altitude	Real
Pressure altitude	Real
Free air correction from altitude	Real
Flags: source of altitude	Integer
L-R meter:	
Spring tension	Real
Beam position	Real
Beam velocity	Real
Cross coupling	Real
Cross acceleration	Real
Long acceleration	Real
Total correction	Real
Vertical acceleration, squared (L-R meter)	Real
Heading A	Real
Heading B	Real
Bell meter:	
counts	Real

As the Bell meter does not provide all the measurements supplied by the L-R meter, if a Bell meter is used alone, the MDF may be made shorter by eliminating unused data fields. Some room should be left for future inclusion of new data fields. The MDF should have a fixed record length and be created so that records may be accessed directly with room for a record every 2 s from the beginning of the track to the end. It must be possible to read and rewrite a record without changing the rest of the file.

Since measured gravity is simply the vertical component of acceleration measured by the meter minus the vertical component of accelerations caused by aircraft motion, some of the information in the MDF would appear to be redundant or extraneous. However, these redundancies can permit the computation of gravity, albeit not to the same precision, despite partial system failure.

Other data, such as the cross and long accelerometer measurements, can be used to correct for poor meter performance and leveling errors in certain cases [8, 9]. All these series are placed, if

available, in a uniform way in the MDF. This is to make the procedure for processing data as uniform as possible while allowing for the use of alternate data sources or performance-based corrections when warranted. The next processing step is then to take the best combination of data sets to compute gravity.

The flight log book and the analog gravity rolls recorded during the flight should be checked to confirm when good data were being taken. This information along with the information obtained about the navigation and altimetry from preliminary plotting and editing determines the times to be used for the MDFs. MDFs should be created to store the data time series for these periods at a time spacing of 2 s per record. When first created, each record contains the time for the record; all other data fields should be blank—the other data are posted later. An algorithm to build the MDFs follows.

```
Program MKMDF

 Get the name of the MDF to be created
 Get the start_time of the flight line for this file
 Get the end_time of the flight line
 Time_spacing = 2 s
 Calculate the size of the MDF
   Number_of_records = (length of flight line (s)) / time_spacing + 3
 Create and open a direct access file of the appropriate size and name
 Time tag the records in the file —
 Time = start_time
 Record_pointer = 1
 Do while time ≦ end_time
   Write record number = record_pointer with time field = time
     and all other data fields set to zero
   Record_pointer = record_pointer + 1
   Time = time + time_spacing
Enddo
Write End-of-Data flag record as last record
Close file
END
```

Figure 3 displays the order for filling the MDF, which is gravity meter data first, radar and pressure altimetry (together) second, and then GPS, third.

Raw Gravimeter Data Processing—SGRAV

The L-R meter outputs recorded in the raw data file must be interpolated to an even time spacing and uniformly filtered before they are placed in the MDF. The L-R meter outputs three critical and six useful data types. The critical values are spring tension, beam position, and cross coupling. These values are critical because the vertical component of acceleration as measured by the meter is calculated from these three. Cross and long accelerations, vertical acceleration (squared), and total correction are useful to correct for off-level problems with the meter and to monitor meter performance. All of these values, except for spring tension, have been filtered internally, so spring tension must be filtered to match. The Bell meter does not give all the readings of the L-R, outputs from the Bell meter are accelerations once per second and status every minute; this is not frequent enough to detect leveling errors in an airborne system.

The algorithm that follows is based on the L-R meter, and contains a simple spike remover based on the L-R meter capabilities and a consistency check using the zero calibration. The zero calibration in raw gravity meter records is supposed to be zero or at most a small voltage. If the recorded zero calibration is not small, the record is assumed to be scrambled and not good. For Bell

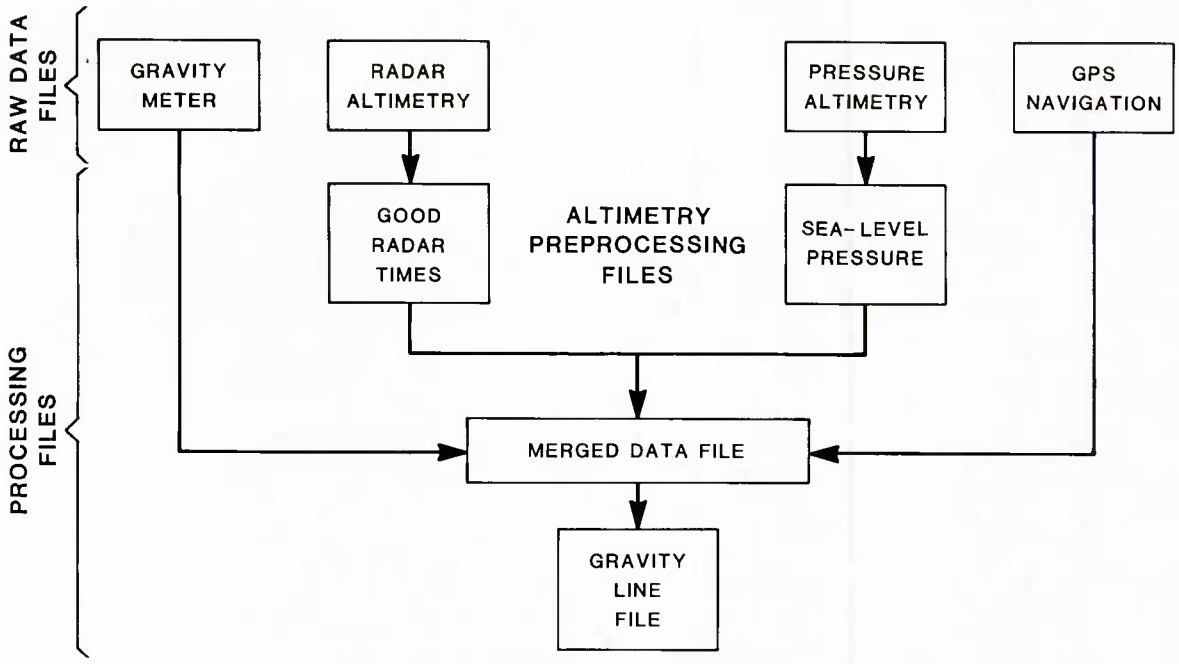


Fig. 3 — Processing file hierarchy and data flow

meter data alone, the measured acceleration should be placed in the field for L-R spring tension, and all the other meter data fields should be set to zero, or not incorporated in the file. If both are used, another field should be employed for the Bell data. The amount of filtering on the Bell data is very slight; the Bell filter introduces so little phase lag that if the Bell data are filtered like L-R spring tension, the other data series filtered to match the L-R can be used to calculate gravity.

The data recorded in the raw gravity meter files must be converted from volts and counts to milligals, the actual scale factors depend on the calibration of the meter. Beam velocity is calculated by the first derivative formula with a 2 s spacing so the conversion factor for beam position should be for V/s to mGal. We assume the programmer will take care of the technicalities of positioning inside files for reads and writes and will move on to new raw data files when encountering the end of a raw data file.

```
Program SGRAV

  Set initial parameters for digital filter appropriate to meter
  Get name of library file and open
  Determine start_time and end_time of gravity line
  Marked_time = start_time
  Read from last raw gravity record before marked_time
          Before_time, meter_values_before:  Meter_values = spring tension,
            cross coupling, beam position, cross acceleration, long acceleration,
            heading A & B, and vertical acceleration squared
  Convert meter_values_before to mGals
  Read from first raw gravity record after or at marked_time:
          time_after, meter_values_after
  Convert meter_values_after to mGals
  Do while (marked_time < end_time)
    Do while (marked_time > time_after)
```

```
    Time_before = time_after
    Meter_values_before = meter_values_after
    Good record = FALSE
    DO while (NOT good record) —
      Read from next raw gravity record
          (Stop if end of raw data)
          Time_after, meter_values_after, zero calibration
      Convert meter_values_after to mGals
      Spike check, consistency check —
      If ( | d beam position / dt | > 500 mGal or
           | zero calibration | > a predefined limit ) then
          Good record = FALSE
      Else
          Good record = TRUE
      Endif
    Enddo
  Enddo
  Linearly interpolate meter values at marked_time
  Calculate beam velocity
  Beam velocity = d/dt (beam position)
  Apply digital filter to spring tension
  Find record for marked_time in library file and write interpolated,
          filtered meter values to the appropriate fields
  Marked_time = marked_time + record time spacing
 Enddo
 Close files
END
```

Altimetry Processing—ALTED, PRPOL, PRSEG, and ALCMB

Altimetry usually requires the most work before it is ready to go into the MDF; this is especially true when operating over land or ice since the radar is then only useful over points of known geodetic height to provide constraints (isobaric slopes) for the pressure altimetry. Radar data over leads in ice, lakes, rivers, runways or other points of known altitude must be abstracted from the radar data files. This is done by using an interactive program that is capable of graphically displaying segments of radar altimetry as defined by time (Fig. 4). This program should have an optional automatic spike remover, one that removes points three standard deviations from the mean; it should also make a file containing the times that define regions of good radar altimetry. These times come from the user by keyboard or cursor position. Subsequent programs use this file to determine whether or not the radar altimetry is good at any given time.

The following is an algorithm for a program to examine and edit radar altimetry files. It is the first step in the processing that precedes the incorporation of the altimetry data into the MDF. To

FILE NAME

PLOT START TIME	038	13:25:01.78
PLOT END TIME	038	14:09:30.49
SECTION START TIME	038	13:52:15.19
SECTION END TIME	038	14:09:30.49

OPTION: (PLOT, DEFINE SECTION, CHANGE FILE, REMOVE SPIKE, QUIT)

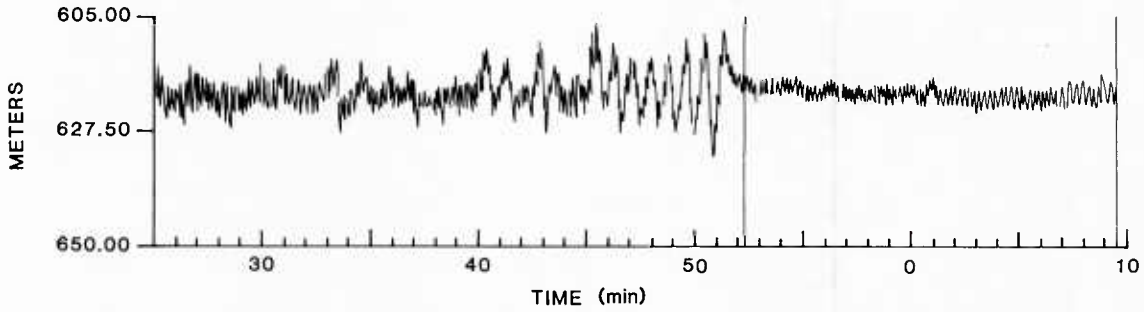


Fig. 4 — Radar altimetry screen editor display. The user has just defined a section of good altimetry between 13:52:15.9 and 14:09:30.49, the program waits for the next option.

determine the times of the data records, the counts recorded with radar altitudes should be added to or subtracted from the record time tag, depending on the relationship to the tagged altitude.

Screen edit algorithm - Program ALTED

```
Get name of good radar sections (output) file
If radar section file exists, read in currently defined sections
Get raw radar altimetry file
Execution loop — Do while TRUE
  Get user processing choice:  plot, define section, change file, remove spike, quit
  If plot then
    Get times of segment to be plotted
    Initialize and set up screen
    Read in data and plot
  Endif
  If define section then
    Read cursor positions for start and end of good section or read definition times
          from keyboard
    Display section limits on screen
    Confirm choice — if approved, add section to list of good sections
    Erase section limits
  Endif
  If change file then
    Blank screen
    Release old raw altimetry file
    Get new raw altimetry file
  Endif
  If remove spike then
  Read cursor positions for start and end of region containing spikes
        or read definition times from keyboard
  Calculate mean and standard deviation of region
  Do for each altitude in region —
```

```
        If altitude is more than three standard deviations from the mean
            then replace it with the average of surrounding altitudes
      Enddo
    Endif
    If quit then
      Sort sections into order merging sections to eliminate overlaps
      Write list to radar section file
      Stop
    Endif
  Enddo — end of execution loop
  END
```

The pressure altimetry computation is drastically affected by the value used for sea-level pressure. Vertical accelerations calculated from pressure data may not be ruined by an erroneous choice of sea-level pressure, since the second derivative is a local process and sea-level pressure usually varies smoothly; but it is crucial to have an accurate absolute altitude for the calculation of the correction to gravity for attenuation from altitude. Therefore sea-level pressure must be accurately determined if pressure altimetry is to be used to augment radar altimetry. During times of good radar coverage, radar altimetry fixes can be used to calculate the sea-level pressure that makes the pressure altimetry agree with the radar altimetry. The two ways to do this are first, to define sea-level pressure by a low-order polynomial over the flight line and second, to define sea-level pressure linearly over short segments. The first method seems to work best when there are a few short periods of good radar for a track, including some coverage at both the beginning and end. The second works best when there are only a few short periods without radar coverage for a track (see Fig. 5). Determining endpoint values for pressure segments is straightforward and can be completely automated. However, if a polynomial is to be used, the best method seems to involve an interactive process in which several low-order polynomials are fit to the data and the best is determined from root-mean-square misfit and personal judgement. In either case, the sea-level constraints should be placed in a file for later use (see Fig. 6). Each survey line should be treated separately so there is one such file for each track, as opposed to one file per flight.

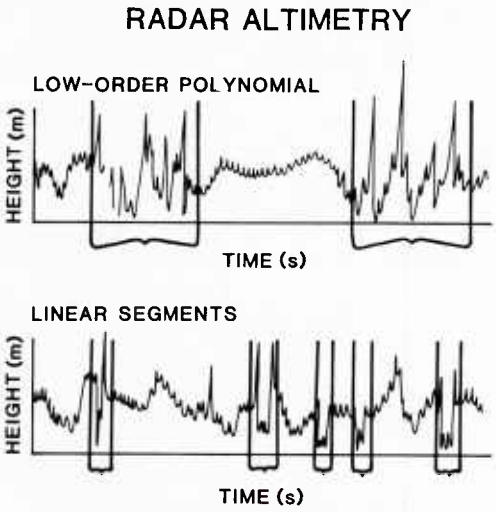


Fig. 5 — Typical sections of radar altimetry, labeled with the method best suited to processing pressure altimetry. Data between braces is suspect and is replaced in the altimetry time series by pressure altimetry.

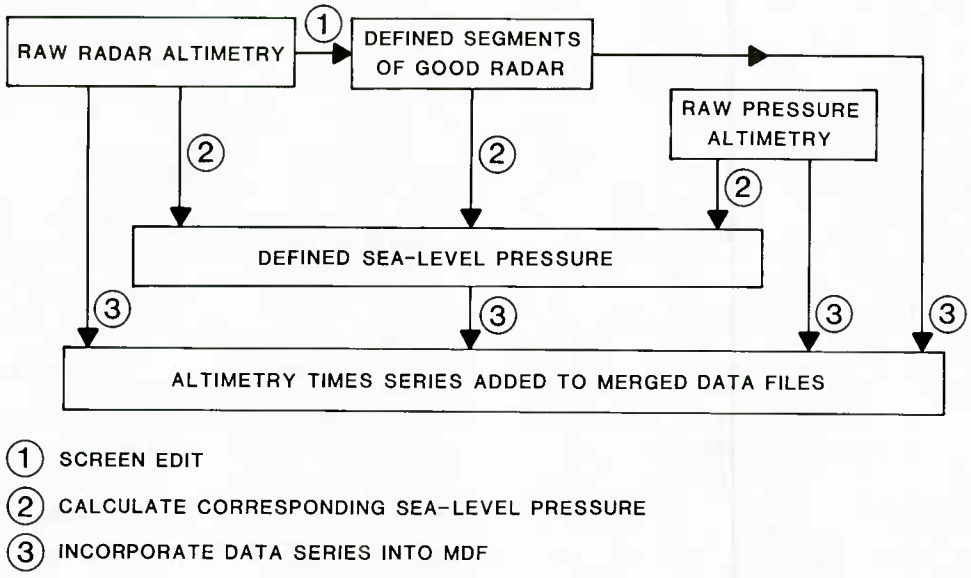


Fig. 6 — Altimetry processing hierarchy and data flow

In both these algorithms it is advantageous to average several raw data points around the time the data value is sought and use the average in place of the raw value. This avoids problems arising from the roughness of the raw data. In the algorithm to define sea-level pressure by a polynomial, we define sea-level pressure as a function of minutes since the beginning of the survey track. Note that pressure readings in volts must be rescaled to millibars before being used to calculate altitude in meters.

Polynomial Algorithm - Program PRPOL

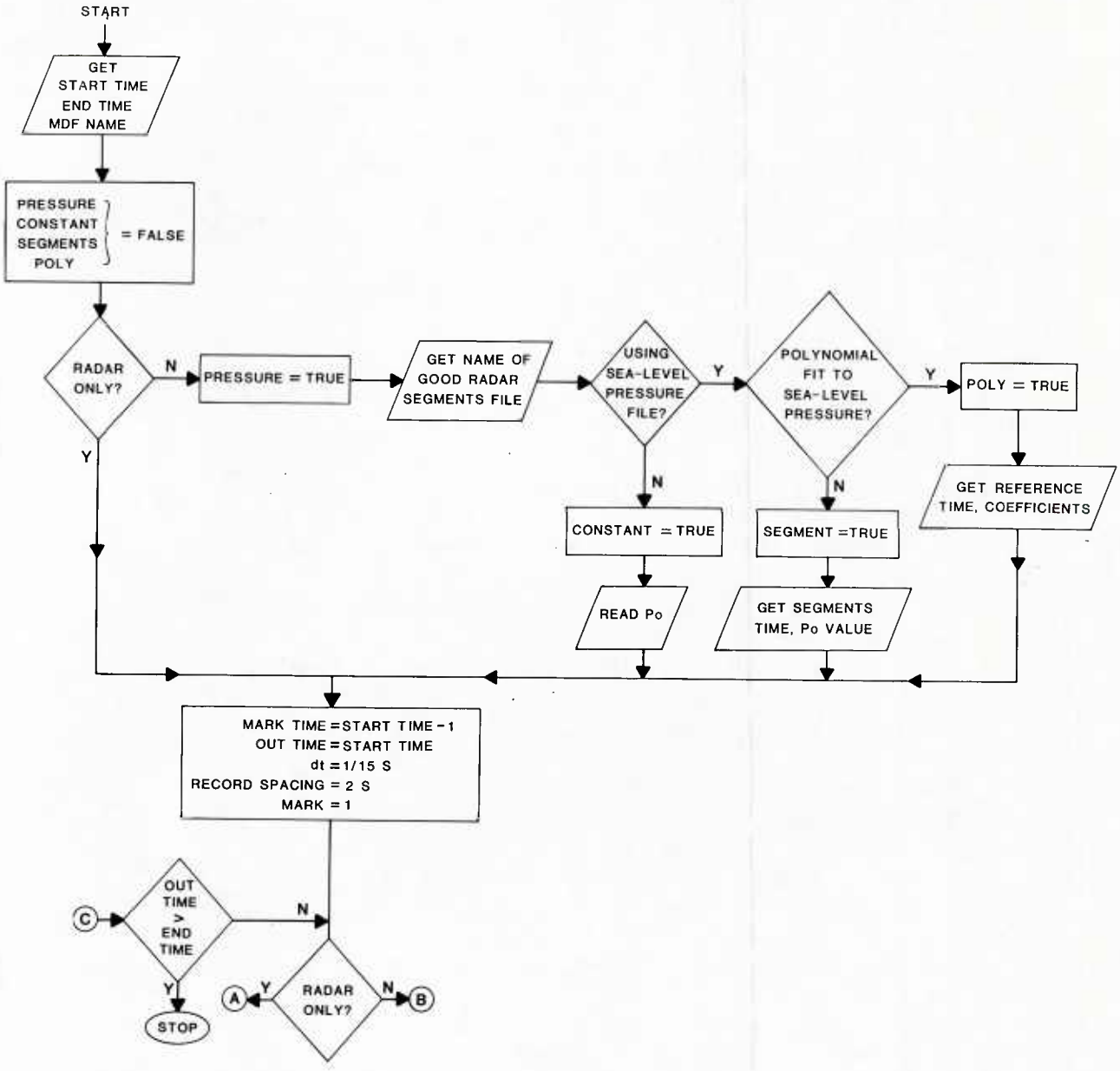
```
Get names of and open the output file, the file with good radar segments defined,
      the raw radar and pressure altimetry files
Read start and end times of good radar segments
Get start and end times of track
Reference time = start time (in minutes),
Write reference time to output file
Working time = start time
Count = 0
Do while (working time ≦ end time)
  If (working time is in a good radar segment) then
    Get radar altitude at working time from raw radar files, convert to meters
    Get measured pressure at working time from raw pressure data files,
      convert to millibar
    Sea-level pressure = (8.4228806e-5 × altitude + pressure^0.190284)^1/0.190284   (5)
    Time (count + 1) = working time − reference time
    Pressure (count + 1) = sea-level pressure
    Count = count + 1
  Endif
  Working time = working time + 1 minute
Enddo
Unhappy = TRUE
Do while (unhappy)
  Plot sea-level pressures vs time
  Read degree of polynomial to fit to data
```

```
    Calculate the coefficients for the polynomial of given degree that best fits,
        in a least squares sense, the sea-level pressure at the abstracted radar data points
    Plot the polynomial on top of previous plot
    Determine if the user is happy with this
  Enddo
  Write the degree and the coefficients of the polynomial in the ouput file
END
```

Linear Segments Algorithm - Program PRSEG

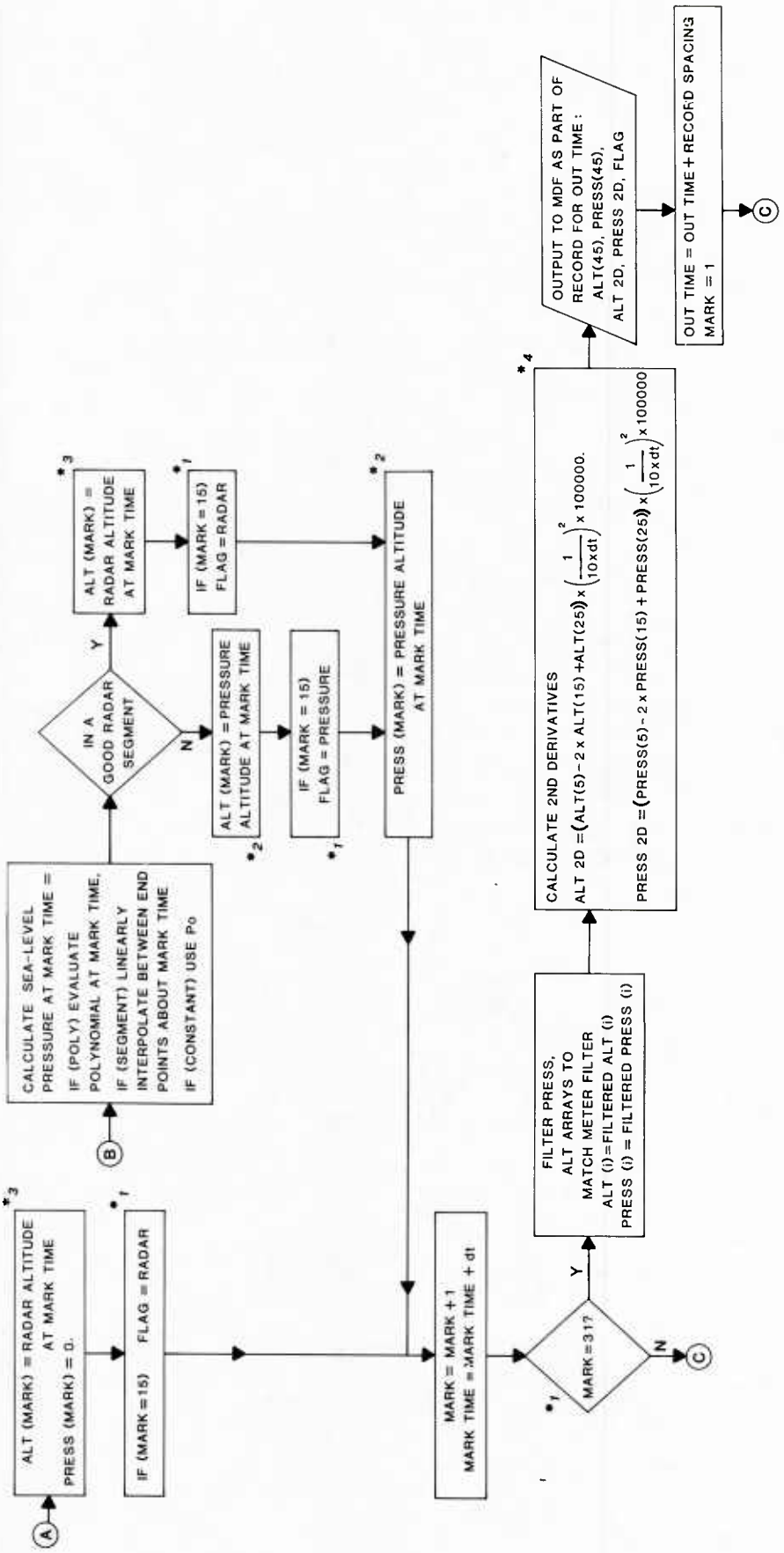
```
  Get names of and open the ouput file, the file with good radar segments defined,
      the raw radar and pressure altimetry files
  Read start and end times of good radar segments
  Get start and end times of track
  If (start time is in a good radar segment) then
    Determine radar altitude at start time from raw radar files
    Determine measured pressure at start time from raw pressure files
    Calculate sea-level pressure
  Else
  Get a sea-level pressure estimate from user
  Endif
  Write start time and sea-level pressure to output file
  For each segment of good radar data, do
  If (start time ≦ start of segment time and start of segment time ≦
      end time) then
    Determine radar altitude at segment start time from raw radar files
    Determine measured pressure at segment start time from raw pressure files
    Calculate sea-level pressure
  Endif
  Write segment start time and sea-level pressure to output file
  If (start time ≦ end of segment time and end of segment time ≦ end time) then
    Determine radar altitude at segment end time from raw radar files
    Determine measured pressure at segment end time from raw pressure files
    Calculate sea-level pressure
  Endif
  Write segment end time and sea-level pressure to output file
  Enddo
  If (end time is in a good radar segment) then
    Determine radar altitude at end time from raw radar files
    Determine measured pressure at end time from raw pressure files
    Calculate sea-level pressure
  Else
  Get a sea-level pressure estimate from user
  Endif
  Write end time and sea-level pressure to output file
END
```

After it has been determined where the radar altimetry is good, where it is not, and what the corresponding sea-level pressure should be; the various altimetry time series that are used to compute gravity can be placed in the MDF. Flags should be set on each record to indicate whether the combined radar/pressure height and vertical acceleration series are from radar or pressure-altimetry data; uncombined times series (such as pressure only) should also be computed at this time. The following flow chart for program ALCMB shows how this is done (Figs. 7(a) and 7(b)). The flow chart is used rather than an algorithm because of the complexity of the program.



(a)

Fig. 7 — Algorithm for program ALCMB to compute altimetry from available sources and for incorporation into the MDF. The numbered asterisks refer to notes in the text.



(b)

Fig. 7 (Continued) — Algorithm for program ALCMB to compute altimetry from available sources and for incorporation into the MDF. The numbered asterisks refer to notes in the text.

Notes concerning the flow chart —

1. The sampling rate is 1/15 s and the record spacing is 2 s. If these values are changed then the constants 15 and 31 in this algorithm must be adjusted accordingly. The relationships between the sampling rate and the spacing are as follows:

   15 = 1/dt

   31 = 1/dt * Record spacing + 1

2. Altitude $= (P_o^{0.190284} - P_m^{0.190284})/8.4228806 \times 10^{-5}$, (6)

   where P_o is sea-level pressure and P_m is pressure measured at altitude.

3. For Altitude see Eq. 1

4. These second derivative formulas include the factors 1/(10 × dt) since the points used are 10 × dt s apart and the factor 100000.0 to convert from m/s^2 to mGals.

GPS Processing—GEOTV

In preparation for filling the MDF, the GPS navigation files should be examined. The navigation positions should be plotted to determine the extent of the navigation coverage and to reveal spikes and other problems within the GPS data. Isolated, anomalous points and all records with two or fewer satellites up should be removed from the GPS files, this may be done with a standard text editor since the GPS files are straightforward ASCII files. Sometimes the navigation will appear to be offset from the track for a segment before returning to the track (see Fig. 8). This is usually caused by a change to a satellite constellation with poor geometry. The offset in position does not usually harm the filtered value of total field for a given latitude, and velocities are usually good once anomalous velocities at the beginning and end of the offset segment are removed.

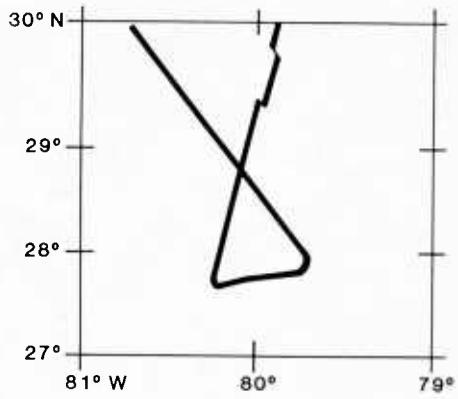


Fig. 8 — Common error in GPS navigation, usually caused by the selection of a satellite constellation with poor geometry. The scale of the track offset is exaggerated.

The data series from the GPS system that should be included in the MDF are latitude and longitude at a given time; velocities north, east, and up; derived accelerations north, east, and up; GPS altitude; calculated Eotvos correction; number of satellites and PDOP; and model gravity field based on latitude. Velocities, altitude, and the model gravity field should be filtered to match the internal gravity meter filter. After gravity has been calculated from these filtered components it is filtered backwards in time with the same filter to restore the values to their navigational positions.

```
Navigation algorithm - Program GEOTV

  Get start and end time
  Get name of and open MDF
  Record spacing = 2 s
  Out time = Start time
  Do while (Out time ≦ End time)
    Interpolate values for latitude, longitude, velocities north,
      east, and up, and altitude
    Determine satellite health − number of satellites and PDOP
    Calculate model gravity field (Eq. 2)
    Calculate total field
    Filter model field value, the velocities north, east, and up,
      and the GPS altitude
    Get radar altitude from MDF
    Calculate the accelerations north, east, and up
    Calculate Eotvos correction (Eq. 3)
    Write latitude, longitude, velocities, accelerations, Eotvos, total field,
      GPS altitude, and GPS health in MDF
    Out time = Out time + record spacing
  Enddo
END
```

Gravity Calculations

The formula for gravity measured at altitude is

$$\text{G measured} = \text{Vertical component of measured acceleration} + \text{Eotvos} - \text{vertical acceleration of plane.} \tag{7}$$

The vertical component of acceleration as measured by the L-R meter is

$$\text{Vertical component} = \text{Spring tension} + \text{beam velocity} + \text{cross coupling} + \text{tie-in constant,} \tag{8}$$

while for the Bell meter it is

$$\text{Vertical component} = \text{Pulse count} \times \text{conversion factor} + \text{tie-in constant,} \tag{9}$$

where the conversion factor converts meter pulse count to mGals. Both meters need an additional constant to tie their output to an absolute reference system. The free air anomaly is given by

$$F = G \text{ measured} + 0.3086 \text{ mGals/m} \times \text{altitude} - \text{Model field}. \tag{10}$$

In the case of an off-level table, L-R meter measurements may be corrected by using the addition L-R meter outputs [9]. The cross, long, and vertical axes of a L-R meter form, very nearly, an orthogonal coordinate system. The local coordinate system of the airplane defined by north, east, and up is also an orthogonal system; the magnitude of the total acceleration should be the same in both systems. In the meter coordinate system, the magnitude (squared) of the accelerations is

$$(TA)^2 = (ST + BV + CC + TI)^2 + (AX)^2 + (AL)^2, \tag{11}$$

where

TA is total acceleration,
ST is spring tension,
BV is beam velocity,
CC is cross coupling,
TI is meter tie-in constant,
AX is cross acceleration from the meter accelerometer, and
AL is long acceleration from the meter accelerometer.

In the navigational coordinate system the total acceleration (squared) is

$$(TA)^2 = (\text{Gravity} - \text{Eotvos} + VA)^2 + (AN)^2 + (AE)^2, \tag{12}$$

where

VA is aircraft Vertical Acceleration (from altimetry),
AN is north Acceleration (from GPS), and
AE is east Acceleration (from GPS).

Therefore,

$$\text{Adjusted Gravity} = ((ST + BV + CC + TI)^2 + AX^2 + AL^2 - AN^2 - AE^2)^{1/2} + \text{Eotvos} - VA, \tag{13}$$

and

$$\text{Adjusted Measured Vertical Component} = \text{Adjusted Gravity} - \text{Eotvos} + VA. \tag{14}$$

when the platform is level,

$$AX^2 + AL^2 = AN^2 + AE^2, \tag{15}$$

so Eq. (13) reduces to Eq. (7) with the substitution in Eq. (15).

All the data necessary to compute gravity are stored in the MDF. Some components are available from more than one source; gravity can be computed in several distinct ways. Choices must be made about which vertical acceleration and altitude to use. For altimetry, radar is preferable to GPS and pressure altimetry when over water. When sea-level barometric pressure is not well-constrained, the derived altitude can vary too greatly from the true altitude to make a good correction possible

1 mbar error in P → ≈ 9 m altitude error → ≈ 2.7 mGals gravity error.

GPS altimetry is not accurate when satellite geometry is bad, and the altitude solution is unconstrained when there are fewer than four satellites. RMS position errors during good geometry are 10 to 15 m for p-code pseudorange solutions which yields an error of 3 to 4 mGals just from the vertical attenuation corrections. Therefore, GPS altitudes are the last choice for computing absolute altitude. The vertical velocities from the GPS can be used for vertical acceleration calculations when neither radar nor pressure are available. However, it will be considerably less accurate than either. Accelerations calculated from pressure altimetry can be good despite sea-level pressure ambiguities resulting from the high-pass filter nature of the second derivative process. So these alternatives may be used when it appears that the radar vertical accelerations are poor or not available in such cases as over land or ice. Vertical acceleration series may be compared to beam velocity, the high frequency component of gravity as measured by the L-R meter. These series should be very similar, and this may indicate which vertical acceleration should be used in the gravity calculation (Fig. 9).

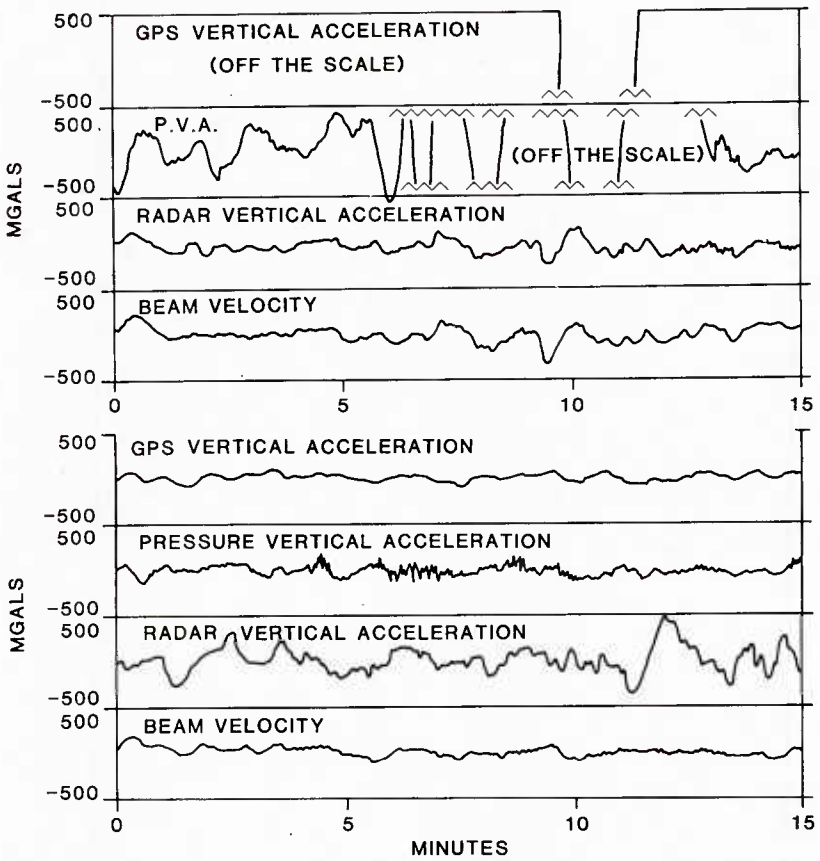


Fig. 9 — Comparison plots of L-R meter beam velocity and available measures of aircraft vertical accelerations. In the upper figure, the vertical accelerations from GPS and pressure are bad, owing to poor satellite geometry and icing of the pressure probe. Vertical accelerations of the radar (here taken over water) are highly correlated with beam velocity and give good values for computed gravity. Below, neither radar (over ice) nor pressure (again during icing conditions) correlates well with beam velocity. The GPS vertical acceleration, however, correlates highly with beam velocity and gives the best computed gravity.

Gravity is calculated from the data in the MDF; it will be recorded, one track per file, in a "Gravity line file." Gravity is computed, recorded, and filtered one track at a time to permit later calculations of systematic track errors. Correcting for such errors increases the accuracy and consistency of the survey. Filtering is easier if records in this file can be accessed directly. Table 8 lists what should be stored in each record of this file

Table 8

Variable name	Type
Time: h, s	2 Integers
Year	Integer
Latitude: deg, min	Integer, Real
Longitude: deg, min	Integer, Real
Raw gravity anomaly (with time lag)	Real
Backwards R-C filtered gravity anomaly	Real
Filtered gravity anomaly	Real
Off-leveling correction	Real
Flags	Integer
Altitude source	2 Bits
Vertical acceleration source	2 Bits
Leveling correction	1 Bit

The following algorithm shows how gravity is calculated for an L-R meter and placed into the gravity line file. Every field above is determined except for the filtered gravity anomaly which is left blank. The raw gravity anomaly is the sum of components from the MDF, this is filtered backwards in time by using the same digital version of the meter's internal R-C filter that restores the anomalies to the proper latitude and longitude. At this stage the gravity usually requires additional filtering: spectral analysis of the gravity shows at what level the survey data is dominated by noise. A filter should be designed based on this cutoff. We find a convolution filter easy to use despite being computationally intensive. For our surveys we have used a linear phase filter design program [10]. When calculated, filtered gravity should be stored in the space reserved for it in gravity line file records. If a convolution-style filter is used, there will be some records at the beginning and end of the line without filtered gravity.

Algorithm to compute gravity - Program GVFGP

```
Get start and end times for gravity track
Open gravity line file
Open MDF
Record spacing = 2 s
Get meter tie-in constant
Determine vertical acceleration sources
Get overall best choice, radar, pressure, or GPS,
Do while there are no more exceptional segments —
   Get alternative source type: radar, pressure, or GPS
   Get start and end times of segment with alternative source
Enddo
Determine altimetry source: get overall best choice
Get, as above, exceptions defined by time
Correct for off-leveling?
```

```
Out time = start time
Count = 1
Do while (out time ≦ end time)
  Read record from MDF
  Determine vertical acceleration source from out time
      VACC = appropriate acceleration from MDF record
  Set vertical acceleration flag to indicate source
  Determine altimetry source from out time
      Height = appropriate altitude from MDF record
  Set altimetry flag to indicate source
  If correcting for off-level problems then
    Calculate adjusted meter measured (Eq. 14)
    Calculate off-level adjustment (Eq. 14 − Eq. 8)
  Else
    Calculate meter measured (Eq. 8)
    Off-level adjustment = 0
    Set adjustment flag to false
  Endif
  Calculate altitude correction
  Calculate anomaly (Eq. 10)
  Backwards filtered gravity = 0
  Filtered gravity = 0
  Write gravity line file record: out time, latitude, longitude, anomaly,
        backwards filtered gravity, filtered gravity, and flags
  Hold_gravity_array (count) = anomaly
  Count = count + 1
  Out time = Out time + record spacing
Enddo
Set up filter constants for backwards filter
Number of points to filter = count − 1
Do i = number of points to filter to 1
  Filter Hold_gravity_array (i)
  Update record # i in gravity line file
  Backwards filtered gravity = filtered Hold_gravity_array(i)
Enddo
Close files
END
```

In addition to the technique described above to correct for off-level errors in gravity, several other methods exist to improve airborne gravity. Cross-correlation analysis can reveal problems in the gravity as well as solutions to these problems. For example, it is very important that data records be accurately time tagged by the acquisition system, this sometimes fails. Timing discrepancies or offsets between the data received from the meter and the vertical acceleration source can be detected through the cross correlation of the beam velocity and radar vertical acceleration time series. These two series should be very highly correlated at zero time lag when filtered in the same fashion as they are in the MDF. A higher correlation away from the zero time lag indicates a timing problem; the lag at which it occurs suggests how it may be corrected. A consistent, higher correlation at a time lag of three seconds radar behind beam velocity indicates the radar time tag is 3 s late. Cross-correlation analysis can also be used to measure L-R meter performance and, in certain cases, to correct for poor performance. The method is described in Ref. 8.

The procedure described above results in a reduced, filtered, line gravity file. This is a single track of reduced gravity. Because of inherent error sources in acceleration, velocity, and altitude measurement, the accuracy of gravity measurement over a region can be enhanced by running survey track lines as a bidirectional grid and by using miss-ties at crossover points to adjust the individual tracks of data. We have used a linear error model for each track and, by least-squares, have determined the slope and bias of each track that minimizes the crossover error [4, 5, 11]. After least-squares adjustment, the remaining crossover error is divided between the 2 tracks at each intersection, and this correction is interpolated between intersections. This results in a smooth contourable data set with minimum random error.

REFERENCES

1. J. M. Brozena, J. G. Eskinzes, and J. D. Clamons, "Hardware Design for a Fixed-Wing Airborne Gravity Measurement System," NRL Report 9000, 1986.

2. J. M. Brozena, "A Preliminary Analysis of the NRL Airborne Gravimetry System," *Geophysics* **49**, 1060-1069 (1984).

3. J. M. Brozena, "An Airborne Gravity Measurement System for Use in the Arctic," Proceedings of the Arctic Oceanography Conference and Workshop, 1985, pp. 30-33.

4. J. M. Brozena and M. F. Peters, "An Airborne Gravity Study of Eastern North Carolina" (publication pending).

5. J. M. Brozena and M. F. Peters, "East Coast Gravity Validation Study" NRL Memorandum Report 5972 (publication pending).

6. Texas Instruments Staff, TI-4100 Owners Manual Appendix A.

7. R. B. Harlan, "Eotvos Corrections for Airborne Gravimetry," *J. Geophys. Res.* **73**, 4675-4679 (1968).

8. L. J. LaCoste, "Crosscorrelation Method for Evaluating and Correcting Shipboard Gravity Data," *Geophysics* **38**, 701-709, (1973).

9. L. J. LaCoste, personal communication, 1981.

10. J. H. McClellan, T. W. Parks, and L. R. Rabiner, "FIR Linear Phase Filter Design Program," in *Programs for Digital Signal Processing* (IEEE Press, 1979) 5.1-1 to 5.1-13.

11. M. F. Peters and J. M. Brozena, "Constraint Criteria for Adjustment of Potential Field Surveys" (publication pending).

DEPARTMENT OF THE NAVY

NAVAL RESEARCH LABORATORY
Washington, D.C. 20375-5000

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300